



API-Documentation

US English, drafted 4/6/2012, Textbroker API version 1.3.1

Table of Contents

An Introduction To Textbroker	1
The Concept and Capabilities of Textbroker's API	3
Requirements to Use WebServices	4
Creating a Budget in a Textbroker Account	5
Web Services Functions	6
Log-In	6
BudgetCheckService - Pulling Budget Information	8
The Functions	8
BudgetOrderService - Ordering and Retrieving Orders	9
Functions	9
BudgetOrderChangeService - Changing an Order	20
Functions	25
BudgetProofreadingService - Creating, Processing and Accepting Proofreading Orders	27
BudgetOrder Status Request Responses	30
Error Messages	31
Update History	33

Textbroker API Documentation

Foreword

Textbroker makes ordering custom-written content and managing those orders simple. With the Textbroker API (Application Programming Interface), you can now implement and manage the entire ordering process automatically or manually from your own web application. Your finished product can be delivered directly to your CMS, blog, forum, or any other online application, without any extra steps.

The API allows you or your software developer to create scripts for your server that will have the ability to send order information to Textbroker, as well as inquire after text and status information. The results will be sent to you in XML-Format in order to work them into your application.

This document describes the structure and functionality of our API to place and manage OpenOrders. For implementation purposes, fundamental programming directions are also provided.

An Introduction To Textbroker

If you are already familiar with APIs in general and Textbroker's API specifically, but haven't yet used the system, this very basic overview will help you better understand our service.

Textbroker is a marketplace for freelance writers and clients needing content. To ensure the best fit for client needs, Textbroker rates all authors by quality, from a minimum of 2 stars to a maximum of 5 stars. The higher the quality ordered, the higher the price per word. The total costs for an article are calculated by the price per word, the number of words, and a fixed per-order fee. The current price list is available at <http://www.textbroker.com/en/client-prices.php>. Textbroker works on a pre-paid basis. In order to place an order, the client must have enough funds in his Textbroker account to cover the costs of the order.

An order includes the following data:

- Category for the article subject matter

A drop-down list of categories is provided.

- Minimum and maximum word count

These settings are unlimited. An author can only submit an order that meets the minimum number of words. The client only pays for the number of words written, up to the maximum word count set by the client. Any words over the maximum are free to the client. You pay at most the maximum number of words set in your order. To ensure that the order is covered, Textbroker earmarks the funds for the maximum number of words ordered. If the author returns less than the maximum, the difference is released to your Textbroker account.

- Processing time

This number sets how many days the author has to write the article once he has accepted it. It is not the total turnaround time or a drop-dead date. If the author does not submit within the time allotted, the order returns to the author pool for another author to choose it. That author also has the processing time to complete the order. A 1-day processing time is recommended for orders of 800 words or less.

- Quality level

The quality level sets the price per word and also prevents unqualified authors from accessing the article.

- Order Title

The title should be short and punchy. This can be just keywords or a full title. The author can enter a different title.

- Order Description

The details of the order should be as specific as possible. The more information included in the order, the better the authors can complete the article to your satisfaction. URLs are welcome, but refrain from including personal contact information like e-mail addresses. Attachments are not available. Please upload your files to a file-sharing site and include the URL for the authors.

- SEO Options

Set keywords and their density in the article, if desired, with this function. Separate keywords and keyword phrases by a comma or a semi-colon. Please see the keyword section for a list of permissible characters in keywords. Enter keywords exactly as they should appear in the article. If you choose a density, it will apply to all requested keywords equally.

When an order is placed, it goes through multiple stages. First, you have a 30-minute cooling-off period. If you have made mistakes in your order, you can easily change them within these 30 minutes. Then Textbroker checks the order. If there are questions or it goes against our Terms of Service, the order may be returned to you. Textbroker staff cannot make any changes to orders placed via the API. When Textbroker accepts the order, it is immediately available to all qualified authors.

If an order is not claimed by an author, it can be deleted. In this case, the funds for the order are released to the Textbroker account.

As long as an author is working on the article, it cannot be deleted.

While the author is working on the order, the only changes that can be made are ones that work in the author's favor: higher maximum word count, lower minimum word count and longer processing time.

If the first draft of a submitted article is not acceptable, you can ask for a revision but not reject the article. The author needs to know what needs to be improved or changed. The number of revisions is unlimited. If the second draft is also not acceptable, you have the additional option of rejecting the article. A rejection requires an explanation. The rejection is then reviewed by Textbroker to see whether the article meets the order requirements and the quality level requested. If Textbroker accepts the rejection, you can choose via the API whether the order should return to the author pool or be deleted.

If Textbroker does not accept the rejection, the article will be accepted on your behalf and the funds will be deducted from your Textbroker account.

Accepted articles can be sent on to proofreading. You can ask for revisions to a proofread article as well. Each text can be sent through proofreading twice.

Two ordering options are available: OpenOrders and TeamOrders. OpenOrders go out to our entire author database. TeamOrders go to a Team of authors that has been created by you. While TeamOrders can be placed and managed via the API, Teams cannot be created or managed via the API. To add or remove authors from a Team, log in to textbroker.com and navigate to "Authors --Teams."

The Concept and Capabilities of Textbroker's API

We use a type of sub-account to your current Textbroker account called a "Budget" as a basis for our API. You can create and manage multiple Budgets within your Textbroker client account.

Both OpenOrders and TeamOrders can be placed in Budgets.

For cost control purposes, you are able to assign a maximum amount that can be spent by each Budget within a given time period, for example, a maximum of \$50 per week. You have full control of the amount of money that can be spent by the person who has access to this Budget. However, it is also possible to assign unlimited funds to your Budget. In this case, the Budget can access as much money as is available in your client account.

You can set the accessibility of each Budget. Third-party access to the Budget via the API does not extend to other sub-accounts or to your Textbroker client account. This is especially important for clients looking to re-sell Textbroker's services (white label) or agencies with multiple managers.

Each sub-client has exclusive access to his or her specific Budget or Budgets and the specific funds that have been deposited by the Budget account holder.

Budget management is implemented through your Textbroker account. All order-related functions can be handled completely through the API without having to log in to Textbroker.

The API allows you to control these account features in your web application:

- Place OpenOrders
- Delete orders
- Approve orders
- Change orders
- Receive notices on the status of your order
- Receive completed articles
- Return articles for revision
- Reject articles
- Send accepted items to proofreading
- Accept, change, reject or delete proofreading orders

Throughout this document, the phrases “**Budget ID**” and “**Budget Key**” will be used for the identification of a given Budget.

The **Budget ID** is a number used to inform your server about changes and assignments through the callback function, which will be described later.

The **Budget Key** is a combination of numbers and letters. This key needs to be communicated from your server whenever you would like to log into a Budget. Simply put, the Budget Key acts as a login name for a given Budget.

Requirements to Use WebServices

The WebServices use SOAP protocol but do not have WSDL (WebService Description Language) files. The Textbroker API is nearly universally acceptable: SOAP clients are available for PHP, PERL, Python und Java. However, clients have experienced challenges when integrating into C# environments.

For example, for the PHP service:

```
https://api.textbroker.com/Budget/loginService.php
```

PHP offers a SOAP client with Version 5.0.1 and above. A simple test shows if the client is available on your server:

```
<?php
if(!class_exists ("SoapClient")){
echo "No SOAP client installed";
}else{
echo "OK";
?>
```

If you **don't** see "OK," there is no SOAP client on your server. Please install one or use the NuSOAP-Toolkit:

```
https://sourceforge.net/projects/nusoap
```

Important: All parameters transferred as a string (text) have to be in UTF-8 coding.

Creating a Budget on www.textbroker.com

Log in to your client account. Go to "Account – Settings" and turn "API" on. You will have a new API tab in the main menu. Mouse over or click on the API tab to see the API navigation bar.

Go to "create a new Budget".

Please have the following ready to create a new Budget:

- E-Mail Address – to keep you updated on your BudgetOrders.
- Password – controls access to this Budget via the API.
- CallbackURL – Your server's URL or the domain that should be notified when the status of an order changes. A program or script for the CallbackURL on your side is not required, but if provided, it allows for a much more efficient way to exchange these messages.

Example: Your CallbackURL will receive the following parameters by HTTP-POST in the case of a completed order: "order_id" (ID of the BudgetOrder), "budget_id" (ID of the corresponding Budget) and Status ("4" or "5" – please see the status list at the end of the document) Additionally, the "proofreading_id" is given (ID of the BudgetProofreading, if available, otherwise "0").

- Budget limits are for a specific time and/or deposit. Month means from the 1st to the end of a calendar month, Year is a calendar year (Example: Setting a Year Budget in July 2010 runs from January 1 through December 31, 2010). Enter 0 to have no limits on your Budgets.

- Test-Mode – Choose "yes" to test your programming code. As long as the account is on Test Mode, the orders will not be sent to authors and you will not be charged. With each request, the order status will evolve until the dummy article reaches "Accepted." You only have access to test orders via the API. You cannot access them via your regular Textbroker account. All sandbox orders are cleaned out every two weeks, so you may encounter an "order doesn't exist" error after a certain period of time.
- Activated – This additional option appears when you want to edit your Budget settings at a later date. This is where you can deactivate a Budget. When you deactivate a Budget, no further work can be commissioned, but current orders will be completed.

After the Budget has been created, you can view your settings. Next to your details, you will see additional information needed to log in:

- Budget ID – The number of the given Budget, which will always be sent on to your callback URL.
- Budget-Key – A key provided by us, which you will need to log in to this Budget.
- Password – Your pre-assigned password for the Budget.
- API URL – The URL you will use to connect to our API.

Web Services Functions

The following is a summary of the functions of the Web Services. Each described area will include relevant programming examples. The following services will be covered:

LoginService

budgetOrderService

budgetOrderChangeService

budgetProofreadingService

Log-In

To start out, necessary variables such as "location" and "uri" (the URL) are inserted, then a SoapClient-Object is created:

```
$options = array('location' =>
'https://api.textbroker.com/Budget/loginService.php',
'uri' => 'https://api.textbroker.com/Budget/'
);
$client = new SoapClient(null, $options);
```

doLogin – Log-In is only possible with this function.

Function "doLogin"

The function requires three parameters:

Salt – “the salt in the soup.” This is vital for Budget Password security and should change with each request. (Entropy for the md5-Method)

Token – an md5-Hash comprised of Salt+Budget-Password: for example, md5("a"."Mypassword") shows "4ac9b0cea3d1c9c95e671bf332dd3aad", where "a" is Salt and "Mypassword" equates to the Budget-Password. Changing the Salt would result in a completely different outcome. The Budget-Password is separate from the password for the client account (Client Password).

Budget Key – the key to the Budget that you want to log in to.

Please be aware that no field can be left blank.

Here is an example:

```
$response = $client->doLogin('a',  
'a9eda88483a97c75aba3434c62009797', 'abc12f6548930fe6ae53b');  
/* or */  
$salt = rand(0, 10000);  
$password = '0123456';  
$response = $client->doLogin($salt, md5($salt. $password),  
'abc12f6548930fe6ae53b');
```

The service’s response lies within the variables \$response. It will be either “true” or “false” (the boolean operator, not a word string). **You will need these three parameters for every inquiry.**

In all other services, instead of "true" or "false," you will receive an associative array (called "HashMap", "Dictionary" or "Map" in other programming languages) of one or more elements. The element "error" will appear in the place of "false". A more detailed description will then be included in the error message.

If the Budget is found to be in Test Mode ("sandbox"), the element "sandbox" with the value of "1" will appear.

Please change the Salt regularly in order to ensure the security of your access information. We reserve the right to refuse Salts that have been repeatedly used or that come after each other too quickly.

BudgetCheckService – Pulling Budget Information

The following function pulls more detailed information, such as use, order type, Test Mode, information on time use, about a certain budget:

```
https://api.textbroker.com/Budget/budgetCheckService.php
```

All functions use the same parameters as `loginService.php` above. The function can be activated manually. If a CallbackURL is in place, the function will update automatically every 30 minutes.

The Functions

Questions regarding your Budget

`getUsage` – The current balance of the Budget.

`isInSandbox` – Conveys whether the Budget is being conducted in Test Mode (array element "**sandbox**" = 1).

`getName` – Reports the name given in the client interface.

`getActualPeriodData` – Gives more exact information about the duration and usage of the Budget. (No "sandbox" element in Test Mode.)

An array with the following elements will be returned:

`usage` – A number with two decimal places. Applies only to accepted BudgetOrders. (Not in sandbox mode)

`sandbox` – Indicates whether sandbox mode is on. "1" is on, "0" is off.

`name` – The name supplied from within the client interface (only with `getName`).

`start` – Start time as a Unix time stamp (integer).

`end` – End of the Budget period as a Unix time stamp (integer).

`left` – Available balance in the Budget (float).

`locked` – The maximum amount being held against all orders in the Budget, to be released upon order acceptance (float).

`max` – The absolute maximum balance allowed for the Budget (float).

If a Budget has no limits, 'start' and 'stop' will be expressed as 0, 'left' is smaller or 0, and 'max' is 0.

A "one-off" Budget (not repeating but limited) has the number 2147483647 as the 'end'.

To calculate the remaining balance, use the max-locked rule, which will show your remaining balance. It can be used securely.

```
$salt = rand(0, 10000);
$password = '0123456';
$response = $client->getUsage($salt, md5($salt.$password),
'abc12f6548930fe6ae53b');
echo $response['usage'] . ' of the Budget have already been
used';

$response = $client->getActualPeriodData($salt,
md5($salt.$password), 'abc12f6548930fe6ae53b');
if($response['error'] == false)
{
    echo 'Until'.date('m/d/y', $response['end']). ' you can
place orders for up to '($response['max']-
$response['locked']).';
}
else { echo $response['error'];}
```

BudgetOrderService – Ordering and Retrieving Orders

Orders are placed, retrieved and deleted through

```
https://api.textbroker.com/Budget/budgetOrderService.php.
```

Here you can also request information regarding the status of your order. The functions can be activated manually. If a CallbackURL is in place, the functions will update automatically every 30 minutes.

Functions

Function
getCategories

getCategories – Delivers an associative array of categories with the respective Category-IDs

Parameters: none, "getCategories()" is sufficient

Return array:

"categories" - Associative array of categories. For example:
"Astrology"=>110, "Telecommunications"=>133, "Finances"=>115

error [optional with mistakes] – Error description (String)

```
$categories = $client->getCategories();  
$auto_cat_id = 0;  
foreach($categories as $name => $id)  
{  
    echo $name. ' has the ID '.$id;  
    if($name == 'Auto')  
    {  
        $auto_cat_id = $id;  
    }  
}
```

create – Generates an OpenOrder.

Function create

This function is only operable if a Budget is currently active.

Parameters:

All functions use the same parameters as loginService.php.

Additional Arrays/Elements:

category_ID – can be identified with the help of "getCategories()"

title – Title of the order, visible to the authors

order description – More exact description of what is expected of the author.

min – Minimum word count required for submission.

max – Maximum word count that will be charged.

stars – Quality levels between 2 and 5 (inclusive).

deadline [optional] – a number between 1 and 10 (days). If no number is given, 1 day will be chosen as a default.

Note [optional] – If you want to leave yourself a note that is displayed on pickUp, there are two parameters: first, a 0 and then your text note.

The functions must be done in the order listed.

When your Budget is 80% full, you will be notified by e-mail.

Return Array:

budget_order_id – Order ID (integer)

error [optional with errors] – Error description (String)

sandbox [in Test Mode] – the value 1 (integer) when in Test Mode

```
$salt = rand(0,10000);
$token = md5($salt.'mypassword');
$budgetKey = 'abc12f6548930fe6ae53b';
$category = 110;//can be researched with getCategories
$title = "Siemens Cell Phone";
$descr = "Which would you choose and why?";
$min = 100;
$max = 200;
$stars = 3;
$deadline = 2; //has to be completed in 2 days
$response = $client->create($salt, $token, $budgetID,
$category, $title, $descr, $min, $max, $stars, $deadline);
if($response['error'] == null)
{
    echo "Order created with the following ID:
".$response['budget_order_id'].";
    $response2 = $client->getStatus($salt, $token, $budgetID,
$response['budget_order_id']);
    echo "\nTyp: ".$response2['budget_order_type'].", Status:
".$response2['budget_order_status'];
}
else
{
    echo $response['error'];
}
```

Function getCosts

getCosts - Delivers an array with cost information

Parameters: Same as log-in data

Expected: number of words => \$word_count and the quality level => \$classification.

Return Array:

classification - Quality level

word_count - Word count

cost_per_word - Cost per word in each quality level

cost_order - Total word costs for the order

cost_order_fee - The processing fee for the order

cost_total - Total costs for the order, including all costs and fees

currency - The currency. Currently, only US dollars are available.

getStatus – Requests the order status

Function

getStatus

Sandbox Mode/Test Mode: With each cycle, the order progresses to the next step in the ordering process to simulate a real order (order placed -> in progress -> complete). Exception: Sandbox-Orders automatically jump from "accept" to ACCEPTED, from "pickUp" to DELIVERED and from "delete" to DELETED.

Parameters:

All functions use the same parameters as loginService.php.

Additionally:

BudgetOrder-ID – the target BudgetOrder ID. It is returned by the "create" function in the element "budget_order_id".

Return Array:

budget_order_status – Short description of the current status (String)

budget_order_status_id – Number indicating the status (see BudgetOrder characteristics) (integer)

budget_order_id –BudgetOrder ID that is requested (integer)

budget_order_created –Unix time stamp of order creation (Integer)

sandbox [optional] – Order is placed in Test/Sandbox Mode = 1 (Integer)

error [optional with errors] – Error description (String)

```
$orderId = 123;//was returned during create()
$response = $client->create($salt, $token, $budgetID,
    $orderId);
$response = $client->getStatus($salt, $token,
    $budgetKey, $orderId);
echo 'Order ID '.$orderId. ', created on
'.date('m/d/y', $response['budget_order_created']).':
'.$response['budget_order_status'];
if($response['sandbox'] != false)
{
    echo '<br/>is a sandbox order and will not be billed.';
}
```

getOrdersByStatus – Lists orders by the current status

Function

getOrdersByStatus

Parameters: Uses **log-in** parameters and:

status-ID – numerical value that identifies the order status, like ACCEPTED = 5 (see also BudgetOrder status codes)

Return-Array:

an array of all Budget-Order-IDs that have reached the status requested. This can return an empty array.

```
$ready_orders = $client->getOrdersByStatus($salt,  
$token, $budgetKey, 4);  
echo "Aufträge, die akzeptiert werden können: " .  
implode(', ', $ready_orders);
```

delete – Deletes a placed order before it is picked up by an author

Function delete

Note: This is only possible when the BudgetOrder is not being worked on.

Parameters: Uses the same parameters as **getStatus**

Return Array:

order_id_deleted – ID of the deleted BudgetOrder (Integer)

sandbox [optional] – Shown when the Order is being run in Test Mode = 1 (Integer)

error [optional with errors] – Error description (String)

```
$salt = rand(0,10000);  
$token = md5($salt.'mypassword');  
$budgetKey = 'abc12f6548930fe6ae53b';  
$orderId = 456789;  
  
$response = $client->delete($salt, $token, $budgetID, $orderId);  
If($response['error'] != false)  
{  
    echo $response['error'];  
}  
echo $response['order_id_deleted']. ' deleted';
```

preview – Preview of the text to be accepted

Function preview

The order has been written and can be reviewed as a basis for the decision to "accept, revise or refuse".

Note: The order must have the status of "READY"

Note: This step is only the precursor to the function "accept". Both functions correspond with the acceptance in the Textbroker client account.

Parameters: Uses the same parameters as **getStatus**

Return Array:

`your_title` – Title of the order as given by the client (String)

`count_words` – Number of written words (Integer)

`author` – Author nickname (String)

`title` – Title of the order as given by the author (String)

`text` – The text (String)

`classification` – Ordered quality level (Integer)

`error` [optional with errors] – Error description (String)

`budget_order_status` [optional with errors] – Status of the order (String)

`budget_order_status_id` [optional with errors] – Status-ID (Integer)

`budget_order_id` [optional with errors] – Requested BudgetOrder-ID (Integer)

`budget_order_created` [optional with errors] – Unix time stamp of order creation (Integer)

```
$salt = rand(0,10000); $token = md5($salt.'mypassword');
$budgetKey = 'abc12f6548930fe6ae53b';
$preview_data = $client->preview($salt, $token, $budgetID,
$orderID);
echo "<form action='".$_SERVER['PHP_SELF']."' method='post'>
<input type='hidden' name='order_id' value='$orderID'>
Title: ".$preview_data['title']."<br>
Text: ".$preview_data['text']."<br>
<select name='classif'>
    <option value='0'>accept, no rating</option>
    <option value='1'>excellent</option>
    <option value='2'>good</option>
    <option value='3'>not bad</option>
    <option value='4'>poor</option>
</select>
Message:<textarea name='message'></textarea>
<input type='submit' value='go'>
</form>";
```

Function
previewHighlightKeywords

`previewHighlightKeywords` – Preview the submitted item with keywords highlighted

Works the same as **preview**, with the extra feature of highlighting keywords included in the keyword portion of the order.

NOTE: The order must be on "READY" status. This is precursor to the function "accept". Both functions correspond with the acceptance in the Textbroker client account.

Function
CopyScape

CopyScape – Automatic comparison of the submitted item against previously submitted items and the web to avoid plagiarism

Function: `getCopyscapeResults($salt, $token, $budgetID, $orderId, $sandbox_response_code)`

This information is necessary for every login.

The following will be expected:

`$orderId` - the order ID, optional (only available in sandbox mode)

`$sandbox_response_code`, an integer 0-2 to reflect 3 possible outcomes

The following will be returned:

answer ['response_code'] - integer - 0 = not available, 1 = no match has been found, 2 = matches have been found

answer ['response_text'] - string - not available, no match has been found, matches have been found

answer ['checked'] – Time stamp of the anti-plagiarism check

answer ['results_count'] - integer – Number of results

answer ['copyscape_results'] - empty Array oder associative Array

Examples:

```
Array
( [response_code] => 0
  [response_text] => not available
  [checked] => 0
  [results_count] => 0
  [copyscape_results] => Array
  ( ) )
Array
( [response_code] => 1
  [response_text] => no match has been found
  [checked] => 1294783944
  [results_count] => 0
  [copyscape_results] => Array
  ( ) )
Array
( [response_code] => 1
  [response_text] => no match has been found
  [checked] => 1306146544
  [results_count] => 3
  [copyscape_results] => Array
  ( [0] => Array
    ( [matching_words_count] => 15
      [copied_from] => http://example-url-xyz.com/a.html
      [matching_words] => ... Lorem ipsum dolor sit amet, consectetur
        adipiscing elit. ... Fusce nec est eget neque tincidunt tempor. ...
    )
  )
  [1] => Array
    ( [matching_words_count] => 14
      [copied_from] => http://example-url-xyz.com/b.html
    )
  )
)
```

```
[matching_words] => ... Lorem ipsum dolor sit amet, consectetur  
adipiscing elit. ... Fusce nec est eget neque tincidunt ... )  
  
[2] => Array  
( [matching_words_count] => 13  
[copied_from] => http://example-url-xyz.com/c.html  
[matching_words] => ... Lorem ipsum dolor sit amet, consectetur  
adipiscing elit. ... Fusce nec est eget neque ... )
```

pickUp – Picks up completed and accepted orders.

Function pickUp

Note: In order to pick up an order, whether in the API or in Textbroker's client area, it must first be accepted. (BudgetOrder-Status = ACCEPTED). Sandbox/Test-Orders go through the "**accept**" function instead.

Automated order acceptance as an answer to our "Callback" (status=READY) is acceptable but not recommended.

Sandbox/Test Mode: A "Lorem ipsum..." Text is delivered with the maximum number of words for orders placed in Test Mode.

Parameters: Uses the same parameters as **getStatus**

Return Array:

your_title – Title given by the client (String)

your_description – Order description given by client (String)

your_note – Notes written by the client (String)

author – Author nickname or ID

min_words – Minimum word count (Integer)

max_words – Maximum word count (Integer)

count_words – Actual word count (Integer)

title – Title given by the author (String)

text – Text content (String)

already_delivered – **0** if the order is being picked up for the first time, **1** if pickUp has already taken place. This lets you know whether the article may have already been used. (Integer)

sandbox [optional] – the Order was placed in Test Mode = **1** (Integer)

error [optional with errors] – Error description (String)

budget_order_status [optional with errors] – Status description (String)

budget_order_status_id [optional with errors] – Status-ID (Integer)

budget_order_id [optional with errors] – Requested BudgetOrder-ID (Integer)

budget_order_created [optional with errors] – Unix time stamp of order creation (Integer)

```
$salt = rand(0,10000);
$token = md5($salt.'mypassword');
$budgetKey = 'abc12f6548930fe6ae53b';
$orderID = 456789;

$response = $client->pickUp($salt, $token, $budgetID,
$orderID);

if($response['error'] == null)
{
    $SQL = "INSERT INTO myDatabase (id,titel,text) VALUES
('','$'.mysql_escape_string($response['title']).'',
'".mysql_escape_string($response['text']).'')";
    //$SQL send to DB
}
elseif($response['error'] != null)
{
    echo $response['error'];
}
Else
{
    echo 'Order Status: '.$response['budget_order_status'];
}
```

Function
pickupHighlightKe
ywords

pickupHighlightKeywords – Picks up complete, accepted items with highlighted keywords

Works just like the pickup function and also highlights any keywords included in the keyword line in the order.

revise – Request a revision

Function revise

Note: The article in question must be "READY", as with acceptance. As soon as a revision is requested, the author is given a 24 hour deadline, and the BudgetOrder shows as being processed.

Parameters: Uses the same parameters as **getStatus**.

Additionally:

Message– This cannot be left blank if the article is being rejected or a revision is requested so that the author knows what needs to be changed. Must be over 50 characters.

Return Array:

result – "OK" if everything went well (String "OK")

order_id_revised – BudgetOrder ID of the requested revision

sandbox [optional] – Shown when the order is being run in Test Mode = 1 (Integer), otherwise 0

error [optional with errors] – Error description (String)

budget_order_status [optional with errors] – Status of the order (String)

budget_order_status_id [optional with errors] – Status-ID (Integer)

budget_order_id [optional with errors] – Requested BudgetOrder-ID (Integer)

budget_order_created [optional with errors] – Unix time stamp of order creation (Integer)

```
$salt = rand(0,10000); $token = md5($salt.'mypassword');
$budgetKey = 'abc12f6548930fe6ae53b';
$answer = $client->revise($salt, $token, $budgetID,
$_POST['order_id'], $_POST['message']);
if($answer['error'] || $answer['order_id_revised'] !=
$_POST['order_id'])
{
    echo $answer['error'];
    echo $answer['budget_order_status'];
}
else
{
    echo $answer['order_id_revised'] . ' has been sent back
for revision.';
}
```

Function reject

reject – Rejects a finished order

You can only reject an order after requesting at least one revision first. This function sends the article to rejection, where a member of the Textbroker staff reviews the rejection.

Parameters: Uses the same parameters as **getStatus**

Return Array:

order_id_rejected – ID of the rejected BudgetOrder (Integer)

sandbox [optional] – Shown when the order is being run in Test Mode = 1 (Integer), otherwise 0

error [optional with errors] – Error description (String)

```
$salt = rand(0,10000);
$token = md5($salt.'mypassword');
$budgetKey = 'abc12f6548930fe6ae53b';
$orderID = 456789;

$response = $client->reject($salt, $token, $budgetID,
$orderID);
If($response['error'] != false)
{
    echo $response['error'];
}
echo $response['order_id_rejected']. ' rejected -
Textbroker reviews refusal';
```

Function accept

accept – Acceptance of finished orders

Note: The given order must be "READY" in order to be accepted. Sandbox orders use this function to accept articles.

Parameters: Uses the same parameters as **getStatus**.

Additionally:

Rating – an Integer between 0 and 4, used when accepting an order.

0 = no rating, but accepts the article.

1 = excellent

2 = good

3 = acceptable

4 = poor

Message to author – Reason for the rating (optional)

Return Array:

result – "OK" if everything went well (String "OK")

order_id_accepted – ID of the accepted order (integer)

sandbox [optional] – Shown when the Order is being run in Test Mode = 1 (Integer)

title [optional, if OK] - Title of the order as given by the author (String)

text [optional, if OK] - The text (String)

author - Author nickname (String)

error [optional with errors] – Error description (String)

budget_order_status [optional with errors] – Status of the order (String)

budget_order_status_id [optional with errors] – Status-ID (Integer)

budget_order_id [optional with errors] – Requested BudgetOrder-ID (Integer)

budget_order_created [optional with errors] – Unix time stamp of order creation (Integer)

```
$salt = rand(0,10000); $token = md5($salt.'mypassword');
$budgetKey = 'abc12f6548930fe6ae53b';
$answer = $client->accept($salt, $token, $budgetID,
$_POST['order_id'], $_POST['classif'], $_POST['message']);
if($answer['error'] || $answer['order_id_accepted'] !=
$_POST['order_id'])
{
    echo $answer['error'];
    echo $answer['budget_order_status'];
}
else
{
    $SQL = "INSERT INTO myDatabase (id,titel,text) VALUES
('', '".mysql_escape_string($answer['title'])."',
'".mysql_escape_string($answer['text'])."'");
    //$SQL send to database
}
```

Function
acceptHighlightKe
ywords

acceptHighlightKeywords – Delivers the complete article with keywords highlighted.

Works just like the accept function and also highlights any keywords included in the keyword line in the order.

getTeams – Selecting Teams that have already been created

Function getTeams

Note: At least one Team must have been created in order to choose a Team.

Parameters:

Same parameters for all logins.

A two-dimensional array is returned:

team_id – Team ID (Integer)

category_id – Category ID (Integer)

team_name - Team name (String)

team_description - Team description (String)

team_price_per_word - Price per word, set by you (Float)

team_order_fee - Textbroker's per-order fees

team_bonus_per_article [optional] = Bonus per article (Float)

team_public - Whether a Team is by invite only or casting call:
1 = casting call; 0 = invite only

team_status - Team status: 1 = active; 0 = inactive (Integer)

time_created - Date and time the Team was created: Unix time stamp
(Integer)

Example on the next page:

```
$salt = rand(0,10000);
$token = md5($salt.'mypassword');
$budgetKey = 'abc12f6548930fe6ae53b';
$response = $client->getTeams($salt, $token, $budgetKey);
echo "<pre>";
print_r ( $response );
echo "</pre>";

Beispielergebnis bei Erfolg z.B.:
Array
(
    [0] => Array
        (
            [team_id] => 1
            [category_id] => 1100
            [team_name] => Automobile
            [team_description] => Schreiben Sie über aktuelle Schlagzeilen aus
            der Automobilbranche
            [team_price_per_word] => 3
            [team_order_fee] => 30
            [team_bonus_per_article] => 0
            [team_public] => 0
            [team_status] => 1
            [time_created] => 0
        )

    [1] => Array
        (
            [team_id] => 2
            [category_id] => 1100
            [team_name] => test2
            [team_description] => Schreiben Sie über aktuelle Schlagzeilen aus
            der Automobilbranche
            [team_price_per word] => 1
            [team_order_fee] => 30
            [team_bonus_per_article] => 0
            [team_public] => 0
            [team_status] => 1
            [time_created] => 0
        )

    [2] => Array
        (
            [team_id] => 3
            [category_id] => 1097
            [team_name] => name of team
            [team_description] => do something
            [team_price_per_word] => 1.95
            [team_order_fee] => 30
            [team_bonus_per_article] => 0
            [team_public] => 0
            [team_status] => 1
            [time_created] => 0
        )
)
oder wenn keine aktiven Teams vorhanden sind:
Array
(
    [error] => There_are_no_active_teams
)
oder bei Scheitern z.B.:
Array
(
    [error] => Falsche Login-Daten
)
)
```

getCostsTeamOrder – Pulling costs for a TeamOrder

Function

getCostsTeamOrder

Note: At least one Team must have been created in order to get costs for TeamOrders

Parameters:

Same parameters for all logins.

Additional:

team_id - Team ID (Integer)

word_count - Number of words

Returns:

classification = 'TeamOrder' (String)

team_id - Team ID (Integer)

word_count - Number of words written (Integer)

cost_per_word - Cost per word (Float)

cost_order - Cost of the order without our per-order fee (Float)

cost_order_fee - Textbroker's per-order fee (Float)

cost_total - Total order costs including per-order fee (Float)

currency - Currency (String)

Example on the next page:

```
$salt = rand(0,10000);  
$token = md5($salt.'mypassword');  
$budgetKey = 'abc12f6548930fe6ae53b';  
$response = $client->getCostsTeamOrder($salt, $token, $budgetKey,  
$team_id, $word_count);  
if(empty($response))  
{  
    echo 'keine Teams angelegt';  
}  
else  
{  
    echo "<pre>";  
    print_r ( $response );  
    echo "</pre>";  
}
```

Successful example

```
Array  
(  
    [classification] => TeamOrder  
    [team_id] => 3  
    [word_count] => 300  
    [cost_per_word] => 0.0195  
    [cost_order] => 5.85  
    [cost_order_fee] => 0.3  
    [cost_total] => 6.15  
    [currency] => Euro  
)
```

Example of an error:

```
Array  
(  
    [error] => Incorrect Login-Data  
)
```

createTeamOrder – Creates a TeamOrder

Function
createTeamOrder

Note: At least one Team must have been created in order to create TeamOrders

Parameters:

Same parameters for all logins.

Additional:

team_id - Team ID (Integer)

order_titel – Order title visible to authors

order_description – Instructions for the authors

words_min - Minimum word count

Words_max - Maximum word count

`working_time` – Amount of time that the authors have to write once they accept. Not a final delivery time. Integer 1-10 (days)

`note` [optional] – If you want to leave a note to be displayed on pickUp, include this function with the parameters 0 and then the text of the note. Notes are not shown to authors and are for client reference only.

Returns:

`budget_order_id` - Order ID of the TeamOrder, when successful (Integer)

or

`error` - An error with the error description (String)

Example:

```
$salt = rand(0,10000);
$token = md5($salt.'mypassword');
$budgetKey = 'abc12f6548930fe6ae53b';
$response = $client->createTeamOrder($salt, $token, $budgetKey,
$team_id, $order_title, $order_description, $words_min, $words_max,
$working_time, $note);
echo "<pre>";
print_r ( $response );
echo "</pre>";
Example of success:
Array
(
[budget_order_id] => 291801
)

Example of errors:
Array
(
[error] => You don't have enough funds for this order
)
```

BudgetOrderChangeService – Changing an Order

Changes to a BudgetOrder are done here:

<https://api.textbroker.com/Budget/budgetOrderChangeService.php>

Special Case: Changes can only be applied to orders that are not finished or deleted (READY, ACCEPTED, DELIVERED, DELETED, REJECTION_GRANTED and ORDER_REFUSED). Changes to an order are allowed while an author is working or during a revision request process, however, these changes must be to the benefit of the author (increased word count or longer processing time).

Functions

Function setSEO

setSEO – sets the keywords and keyword phrases and the minimum and maximum density

Parameters: Uses the same parameters as **getStatus** and also

keywords – (UTF-8-coded) List of keywords and keyword phrases that should be in the article. Keywords and phrases should be separated by a comma or a semi-colon. Spaces between phrases are allowed. Only UTF-8 letters, numbers and the following characters are acceptable: & (ampersand), ‘ (apostrophe), - (minus sign), % (percent) and . (period). If there are any incorrect characters, they will be replaced with a space. (String)

min – [optional] the minimum number of times that each phrase should occur in the article. Applies to all phrases requested. (Number).

max – [optional] the maximum number of times that each phrase should occur in the article. Applies to all phrases requested. (Number).

Return Array:

“order_id_changed” - when successful, the ID of the changed order

warning [optional – occurs when incorrect characters are entered] – A description of which keyword phrases were changed and how they were saved. Example: This keyword(s) “FF/Z5, Oh!, 66€, \$_[key]” was changed to “FF Z5, Oh, 66, key” (String)

error [optional – occurs when errors occur] – description of the error (String)

```
$salt = rand(0,10000); $token = md5($salt.'mypassword'); $budgetKey = 'abc12f6548930fe6ae53b';  
$min = 10;  
$max = 100;  
$keys = "Easter; egg, rabbit, 2011.04.24";  
$answer = $client->setSEO($salt, $token, $budgetID, $orderID, $keys, $min, $max);  
if($answer['error'] != false)  
{  
echo $answer['error'];  
}  
else  
echo 'Keywords are set.';  
if($answer['warning'] != null)  
{  
echo ' <span class=\'warning\'>'. $answer['warning']. '</span>';  
}
```

changeWorkTime – Changes the allowed processing time

Function

changeWorkTime

Parameters: Uses the same parameters as **getStatus** and

workTime – Processing time expressed in days >=1 und <=10

Return Array:

order_id_changed – ID of the successfully revised order

error [optional with errors] – Error description (String)

```
$salt = rand(0,10000); $token = md5($salt.'mypassword');  
$budgetKey = 'abc12f6548930fe6ae53b'; $workingTime = 5;  
$answer = $client->changeWorkTime($salt, $token, $budgetID,  
$orderID, $workingTime);  
if($answer['error'] != false)  
{  
    echo $answer['error'];  
}  
else  
{  
    echo $answer['order_id_changed'].' changed';  
}
```

Function

changeWordsCount

changeWordsCount – Changes the requested minimum and maximum word length

Parameters: Uses the same parameters as **getStatus** and

min – the minimum number of words required

max – the maximum number of words acceptable

Return Array:

order_id_changed – ID of the successfully revised order

error [optional with errors] – Error description (String)

```
$salt = rand(0,10000); $token = md5($salt.'mypassword');  
$budgetKey = 'abc12f6548930fe6ae53b'; $min = 10, $max = 100;  
$answer = $client->changeWordsCount($salt, $token, $budgetID,  
$orderID, $min, $max);  
if($answer['error'] != false)  
{  
    echo $answer['error'];  
}  
else  
{  
    echo $answer['order_id_changed'].' changed';  
}
```

BudgetProofreadingService – Creating, Processing and Accepting Proofreading Orders

Proofreading can be ordered in any active Budget that is not in test mode. Proofreading orders have a set processing time; the costs are calculated by the length of the text, so creating orders is almost automatic. Changes are not

possible or necessary. There is no test mode for proofreading orders. Proofreading orders have the same conditions as other BudgetOrders. Proofreading is only available for accepted orders.

Proofreading orders are created, accepted, revised and rejected with:

```
https://api.textbroker.de/Budget/budgetProofreadingService.php
```

Most functions are the same as regular BudgetOrders (OpenOrders) from BudgetOrderService.php. The parameter OrderID refers to the proofreading order ID, not the original BudgetOrder/OpenOrder. The return array consists of the same elements, but the names all have „proofreading“ instead of „order,“ „proofreading_id_rejected“ returns for „order_id_deleted“ in the delete function and „budget_order_status“ becomes „budget_proofreading_status“.

The callback has the following changes: The ID of the proofreading order is not delivered in „order_id“ but in „proofreading_id“. „order_id“ is the number 0. All other parameters are the same.

The following functions work analog to BudgetOrderService:

```
preview, getStatus, getOrdersByStatus, delete, accept,
pickUp, revise and reject
```

They use the same parameter as budgetOrderService.php. The function accept delivers a HTML-formatted text that shows the changes from the original. The function pickUp deliverse the changed text.

The functions with significant changes are described below.

Function create

```
create($salt, $token, $budgetKey, $orderId, $instructions,
$title, $text)
```

The first three parameters correspond to the login data. The orderId mentioned in this function, the ID of the BudgetOrder (OpenOrder) to be proofread. This BudgetOrder does not have to belong to the same Budget, but it does have to be accepted.

`$instructions [optional]` – Instructions to the proofreader (String)

`$text [optional]` – the proofread text, if not the same as the original (String)

`$title [optional]` – the proofread title, if not the same as the original (String)

Return Array:

'proofreading_id_created' – ID of the proofreading order (BudgetProofreading) (integer)

'error' – Description of the error that prevented order creation (String)

getCosts – Delivers an array with cost information

Function getCosts Parameters: Log-in parameters.

Expected elements: the text that should be sent to proofreading => \$text

Return Array:

word_count – number of words in the original document

cost_per_word – the costs per word

cost_total – the total cost of proofreading

currency – the currency

Function preview `preview($salt, $token, $budgetKey, $orderId)`

Works the same as a BudgetOrder, but the text in the array element 'text' (but not the title), shows the differences between the original and corrected versions. The result is HTML formatted and should be shown to the user.

Function `acceptHighlightKeywords` – Accepts the article with keywords highlighted

`acceptHighlightKeywords`

Corresponds to the function **acceptHighlightKeywords** from the **BudgetOrderService**.

Function `pickupHighlightKeywords` – Delivers the complete article with keywords highlighted.

`pickupHighlightKeywords`

Corresponds to the function **pickupHighlightKeywords** from the **BudgetOrderService**.

Responses to BudgetOrder Status Requests

From the creation of an article to its final deletion, the following indicators are used:

0 = INTERNAL ERROR

You have placed a BudgetOrder which has not been saved correctly. Please contact support.

1 = PLACED

Your BudgetOrder has been placed and saved correctly.

2 = TB_ACCEPTED

Your BudgetOrder has been processed by Textbroker and is visible to authors.

3 = INWORK

Your order is being written. With this status, a BudgetOrder can no longer be deleted.

4 = READY

Text from the author has been completed and has passed through CopyScape. Your BudgetOrder is waiting to be reviewed. Ownership rights have not yet been transferred to the client. Notification of this status will be delivered to your CallbackURL. An OrderID that is identified as BudgetID with Status 4 is ready to be reviewed.

5 = ACCEPTED

Text has been accepted by the client. All ownership rights have been transferred; the final version of the text can be picked up.

The callbackURL is used to set this status. An OrderID that is identified as BudgetID with Status 5 is ready to be picked up.

6 = DELIVERED

Text has been delivered (particularly important as a control function, so that you don't pick up and use the same article twice)

7 = DELETED

Client has deleted the BudgetOrder (this is possible as long as the order is not being written by an author).

8 = REJECTION_GRANTED

Textbroker has approved the rejection. From here you must decide whether you want to place the order again or delete it.

9 =ORDER_REFUSED

The order could not be released to the authors due to errors in the order description. In this case, you will need to revise your description before continuing.

10 =WAITING

The BudgetOrder is waiting for actions from the client, the author, or Textbroker (Example: when a rejection is awaiting approval from Textbroker).

Possible Error Messages

You may receive the following error messages. The percent signs are variables that will be completed by our system.

'Bad parameters' or 'Missing or incorrect value entered' – One or more parameters are missing or filled in incorrectly.

'%id% is an incorrect category-Id' – The given ID does not match any of our categories. You can find a category with the help of "**getCategories**".

'The Budget %id% is inactive' – The Budget is inactive. No orders can be placed for this Budget, but they can be completed and accepted.

'Choose a quality level >=2', %val% needed' – Incorrect value for an OpenOrder quality level. Possible levels are 2, 3, 4 and 5.

'Not enough funds available' – The client account is not funded or all funds are bound to current orders. The order can therefore not go through.

'Too many orders placed. Budget is limited to already existing ones' (X,Y requested, Z,Z available) – Too many orders have been placed through this Budget. In order not to exceed the Budget limit, this order has been refused. If current orders are completed without using the entire word count, enough funds could possibly become available.

'Budget has been exhausted' – The Budget has been depleted. Further use is not possible. Upon request, the current BudgetPeriod can be closed (advance the end time) by Textbroker within the client account and a new one (from this point in time) can be opened.

'Order does not exist' – Incorrect BudgetOrder-ID. Either you have used a Sandbox-Order ID in the real interface or vice versa, the temporary Sandbox-Order has been erased by the Administrator (after several hours/days), or the given order does not belong in the given Budget.

'Order is being processed. Deletion currently not possible. %time_lock%' – An author has picked up the order. It is not possible for you to erase it at this time.

'Order not yet finished or already accepted' – The order is not yet complete or was already accepted. Same characteristics as with "**pickUp**" as long as the BudgetOrder in question does not yet exist: the same are delivered as with "**getStatus**" in order to understand possible errors.

'Invalid login-data' – The login information is incorrect (Salt, Token, BudgetKey are missing or are incorrect).

'You don't have the rights to this proofreading order.' – The user can only review proofreading orders for Budgets that they have the rights to.

'This proofreading order is not yet ready to be accepted.' – The status of the BudgetProofreading (proofreading order) does not allow you to accept it. (See 'budget_proofreading_status')

'No proofreading order with this ID.' - Proofreading order not found.

This proofreading can't be sent back to the author yet.' - The BudgetProofreading is not in a status that allows it to be sent back to the author.

'This proofreading order is not ready to be rejected.' - The BudgetProofreading status does not allow it to be rejected.

'Looking for %id% in OpenOrders failed' – The BudgetOrder from which the BudgetProofreading should be created doesn't appear to exist.

'This isn't your order.' - The BudgetOrder doesn't belong to you, and therefore you can't create a BudgetProofreading.

'You must accept the order before you can send it to proofreading.' – The BudgetOrder isn't accepted yet. You can only ask for proofreading after you accept the order.

'The proofreading order could not be created.' - An internal error prevented the BudgetProofreadings from being created. Please contact support.

'Proofreading not deleted or money not unlocked in budget' – The deletion of the BudgetProofreadings did not occur or the funds were not released. Please contact support.

'The article is being proofread. You cannot delete the order at this time.' – The item is currently being proofread and is not available for deletion.

Should you receive an error message not covered here, please consult our Textbroker support staff.

Update History

Version 1.0 on Dec. 15, 2009

- Mention of UTF-8 entry added (page 5).
- For the preview function, the return array "author" was added (page 14).

Version 1.1 on 15.02.2010

- "author" was added to the return array of the function `accept` (Page 16).

Version 1.2 on 19.12.2010

- Typographical errors corrected.
- Function `getOrdersByStatus` added (Page 19).
- E-Mail with a warning when a Budget was 80% depleted added.
- `BudgetProofreadingService` added (start Page 22).
- New possible error messages added (Page 24).

Version 1.2.1 on 22.12.2010

- New `create` function in `BudgetProofreadingService` (Page 22).
- Change in Callback parameter (new: "proofreading_id") (Page 22).

Version 1.2.5 on 04.01.2011

- Function `setSEO` added (Page 20).

Version 1.2.6 on 27.01.2011

- Function `getCosts` added in `BudgetOrderservice` (Page 11).
- Return array "author" added in function `pickUp` (Page 14).
- Function `getCosts` added in `BudgetProofreadingService` (Page 24).

Version 1.2.7 am 25.05.2011

- Function `CopyScope` added in `BudgetOrderService` (Page 17).
- New structure of `BudgetOrderService` – follows the logical sequence of the system.

Version 1.3.0 on 19.10.2011

- Function `getTeams` added in `BudgetOrderService` (Page 20).
- Function `getCostsTeamOrder` added in `BudgetOrderService` (Page 22).
- Function `createTeamOrder` added in `BudgetOrderService` (Page 23).
- Function `setSeo` moved to `BudgetOrderChangeService` (Page 24).

Version 1.3.1 on 2.4.2012

- Function `pickUpHighlightKeywords` added to `BudgetOrderService` and `BudgetProofreadingService` (page 17 and 28).
- Function `previewHighlightKeywords` added to `BudgetOrderService` (page 16)
- Function `acceptHighlightKeywords` added to `BudgetOrderService` (page 19 and 28)
- Added information on `TeamOrders` (page 3)
- Corrected „message“ field name for revisions and rejections (page 17)
- Clarified update processes for `BudgetCheckService` and `BudgetOrderService` (pages 8 and 9)